

Amazon SageMaker

Andrew K Jang

Naga Venkata Sai Rama Krishna Rohan Avireddy

Shrey Srivastava

Outline

- Introduction to Amazon SageMaker
- Core Features and Advantages
- Built-in Algorithms and Supported Endpoints
- Distributed Training Deep Dive
- Use Cases and Cost Structure
- Advanced Metrics and Optimization Tips
- Real-Time Model Monitoring
- Hybrid Deployment Strategies
- Integrating SageMaker with MLOps
- Case Study: Accelerating NLP with SageMaker
- Expanding SageMaker Applications
- Live Demo Preview
- Challenges and Future Directions
- Key Takeaways and Q&A

Overview

What is Amazon SageMaker?

- A suite of tools and services provided by Amazon in assisting with the creation, training, and deployment of machine learning models using AWS's cloud resources.
- Machine learning requires many moving parts
- Sagemaker aims to provide a collaborative end-to-end machine learning platform.

3

The Machine Learning Pipeline

Data Preparation

- Data Cleaning
- Data Aggregation
- Data Analysis and Transformation
- Data Validation
- Feature Engineering
- Data Splitting

Model Creation

- Model Building
- Model Training
- Model Validation
- Scaled Training

Rollout

- Deployment
- Serving
- Monitoring, Logging, Explainability
- Visualization

4

History of SageMaker

5

2019: SageMaker Studio

2020: SageMaker Pipelines, SageMaker Distributed, SageMaker Data Wrangler, SageMaker Model Registry

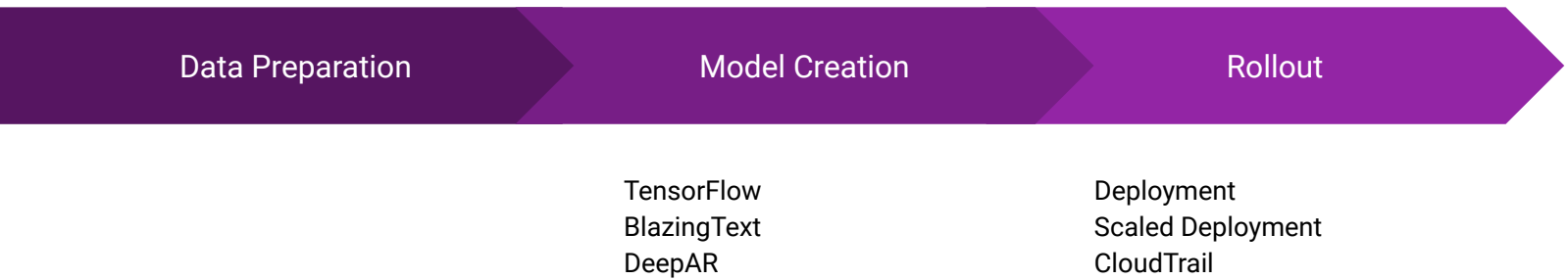
2021: SageMaker Canvas, SageMaker Endpoints

2022: SageMaker Collaboration, Geospatial Capabilities

2023: New SageMaker Experience, Code Editor

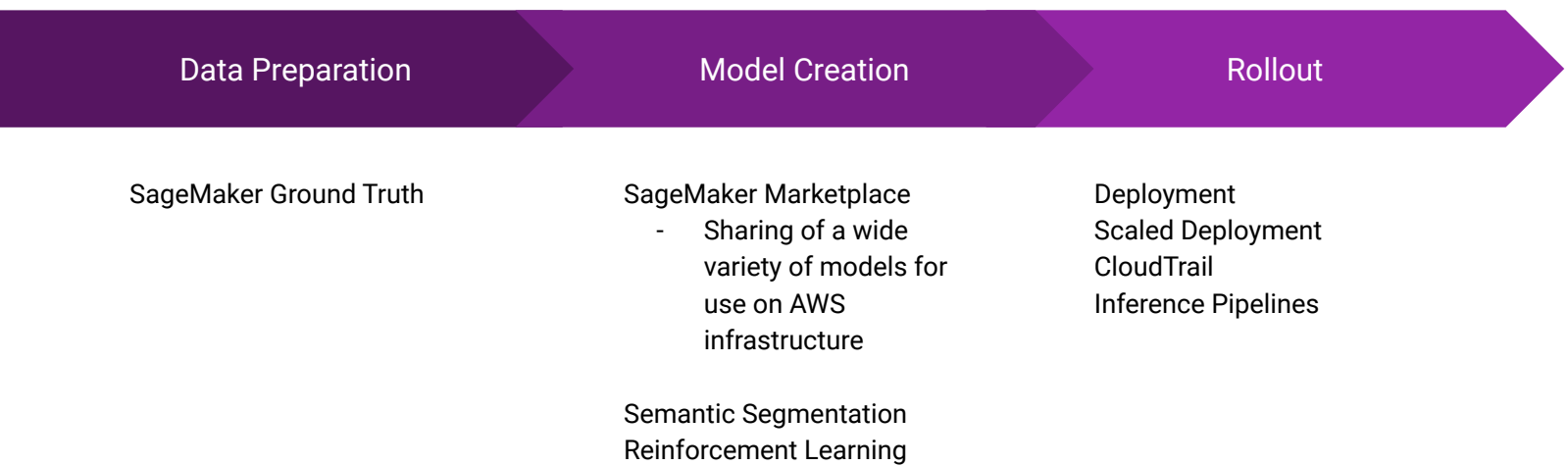
6

Launch of SageMaker - 2017



7

SageMaker Neo, SageMaker Marketplace - 2018



8

SageMaker Studio - 2019

Data Preparation

Model Creation

Rollout

SageMaker Ground Truth

9

Main Feature

Some Features of SageMaker

- **Built-in Algorithms:** Includes XGBoost, Linear Learner, and K-Means for diverse ML tasks.
- **Training, Tuning & Deployment:** Automated tuning and scalable endpoints for real-time, batch, and asynchronous inference.
- **Notebook Instances:** Managed Jupyter notebooks for development.
- **SageMaker Pipelines:** Automates end-to-end MLOps workflows.
- **Integrations & Framework Support:** Works with AWS services (S3, Lambda, Athena) and frameworks like PyTorch, TensorFlow, and Scikit-Learn.

11

JupyterLab Integration

What is JupyterLab?

- Interactive development environment
- Launched with a diverse array of instances
- Run multiple individual Jupyter notebooks
- Develop and test scripts for data processing or machine learning.
- With SageMaker, provides a shared collaborative workspace on the cloud.

12

Creating a JupyterLab Instance for Model Training

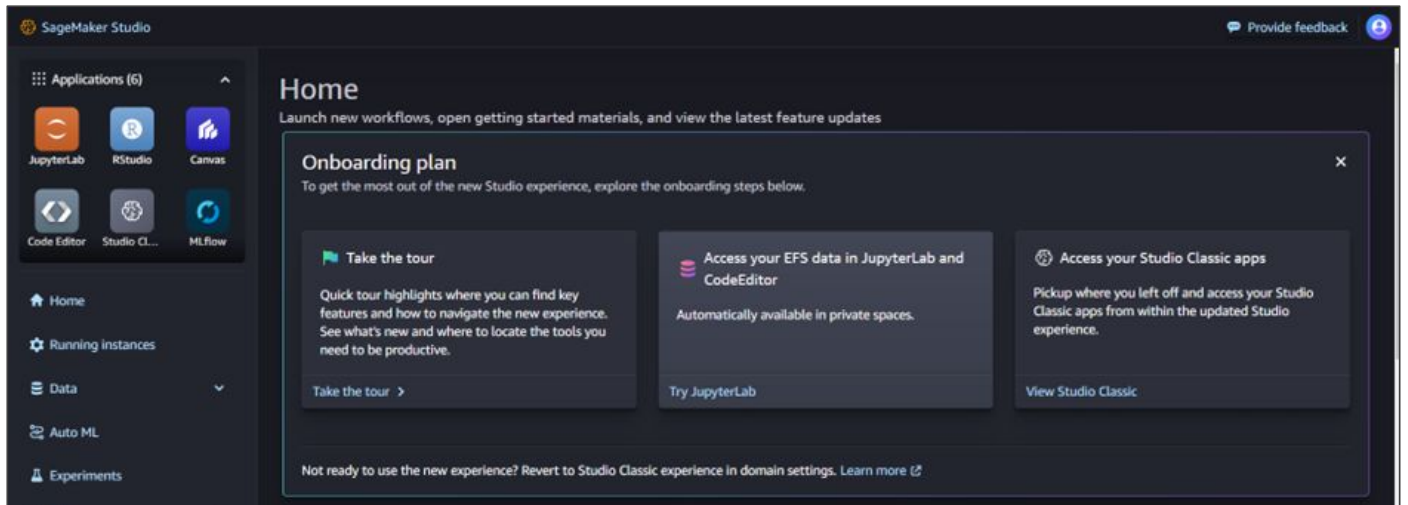
Creating a SageMaker Domain

The screenshot displays the SageMaker Domains console. At the top, there is a 'Domains (3)' section with a search bar and a 'Create domain' button. Below this is a table listing domains. The first domain is 'QuickSetupDomain-20241129T114579' with ID 'd-emp06wlrwp', status 'InService', and creation/modification dates of 'Nov 29, 2024 19:45 UTC' and 'Nov 29, 2024 19:51 UTC' respectively.

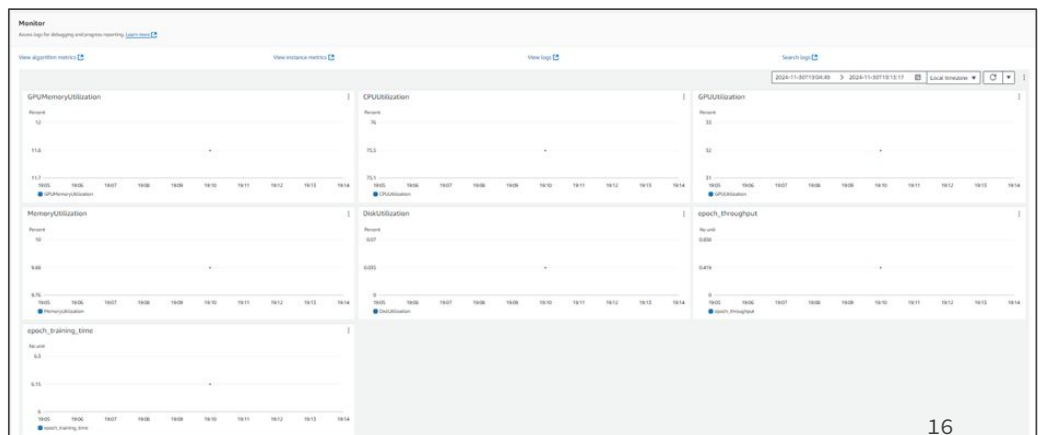
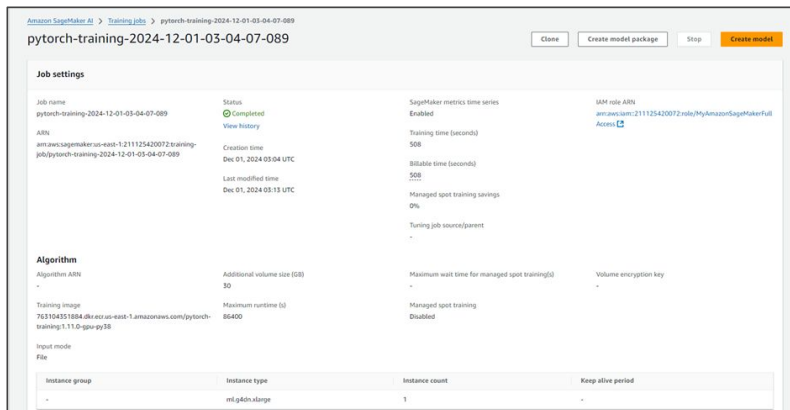
Below the domain list, the 'QuickSetupDomain-20241129T114579' details are shown. Under 'Domain resources', there are tabs for 'Applications', 'Jobs', and 'Endpoints'. The 'Applications' tab is active, showing a search bar and a 'Status: Ready' indicator. A table lists applications:

Name	App type	Status	Instance	User profile	Space Type	EBS Volume(SB)	Created On
ku5562-cloud-computing	JupyterLab	Ready	m13.medium	default-20241129T114579	Shared	5	11/30/2024, 5:12:27 PM

Launching SageMaker Studio



15

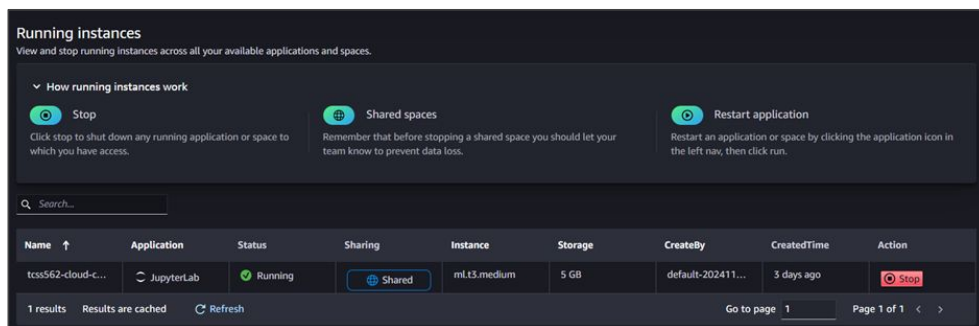


16

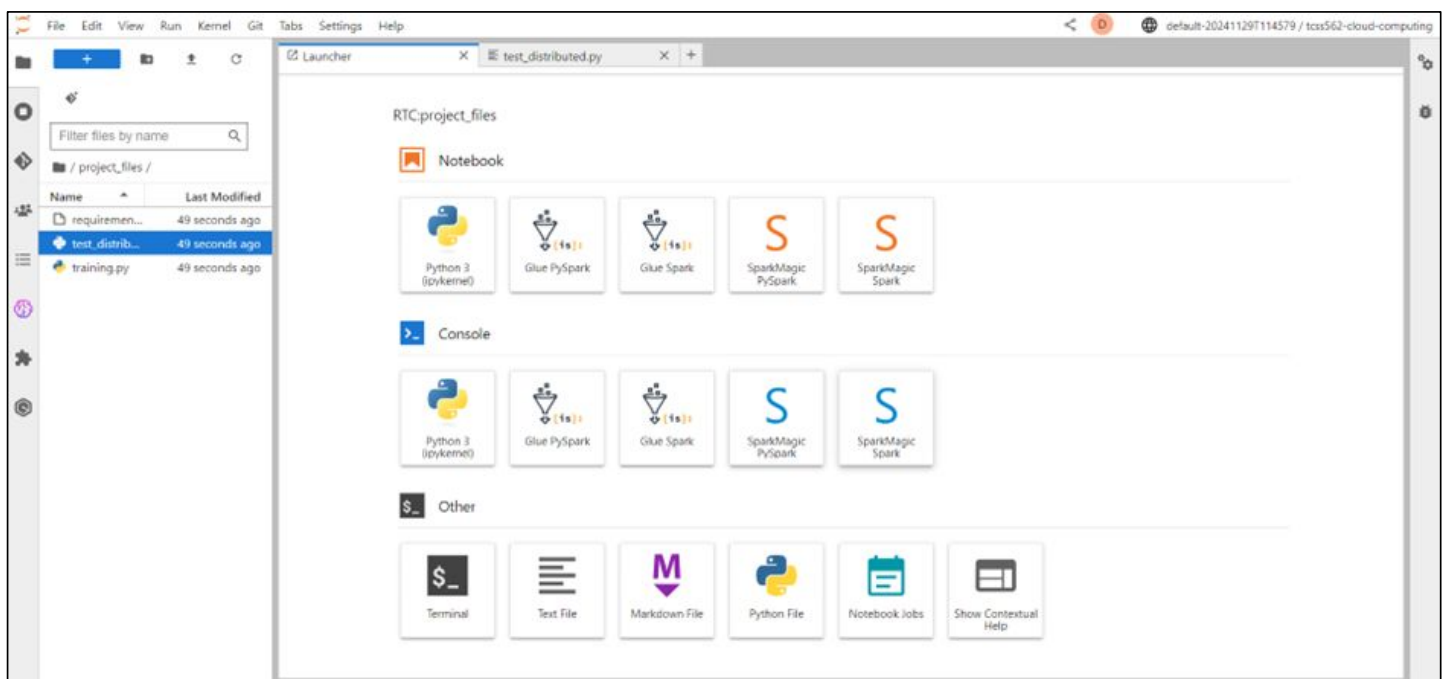
Create JupyterLab Instance

Key Features:

- Shared Workspace
- Comes with pre-configured environment
- Code Editor - development environment
- Jupyter Notebooks



17



18

Example Use Case: Distributed Training

Introduction to Distributed Training

Challenge: Training models with Billions of parameters on a single system -

Challenge: Massive datasets (e.g., terabytes of image or text data) cannot fit into the memory of a single machine.

Challenge: Long training cycles on high-end instances can cost tens of thousands of dollars.

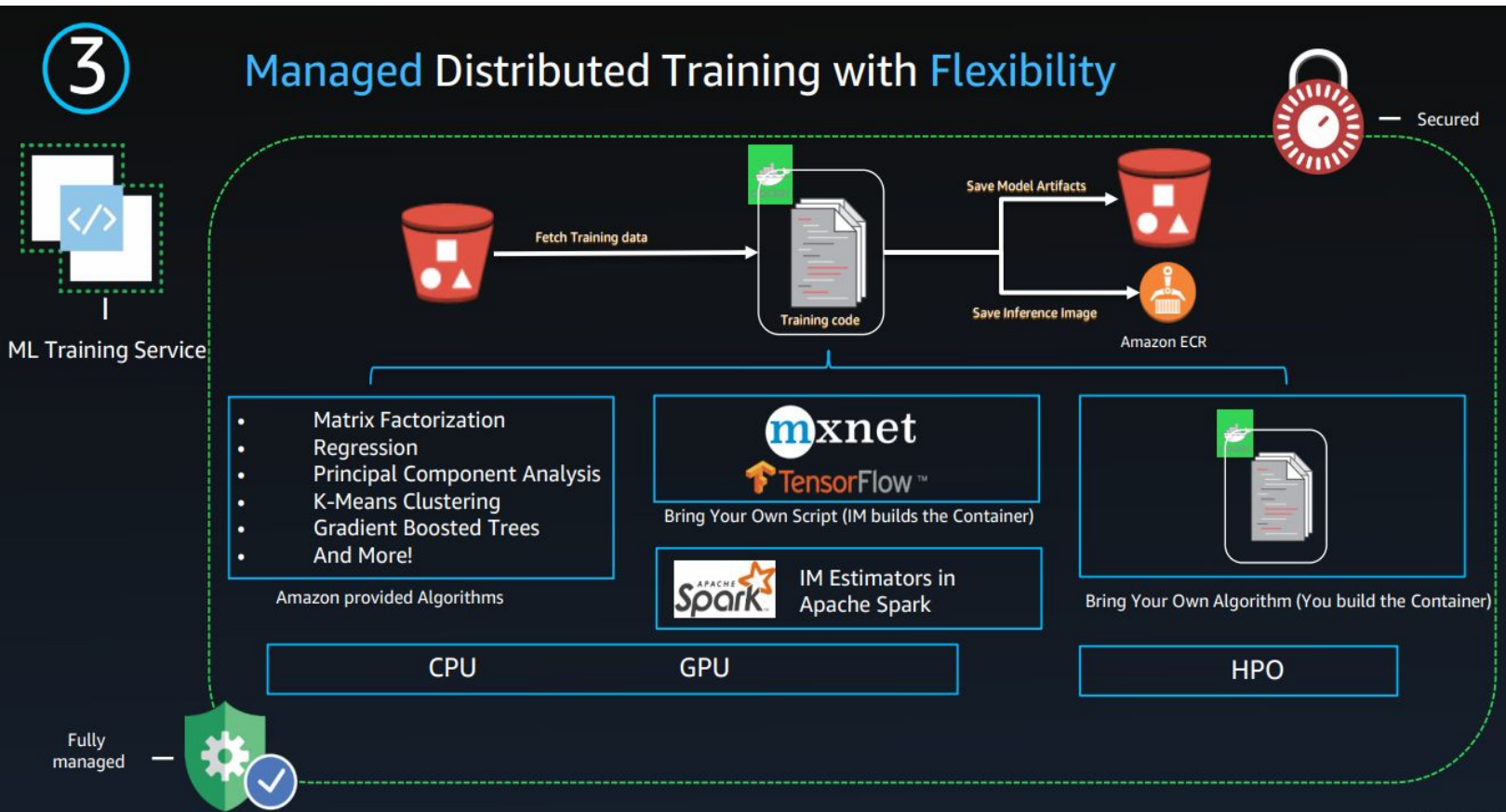
Challenges in Distributed Training (Traditional Setup)

High barriers to entry:

- Setting up and maintaining clusters.
- Managing data distribution and model synchronization.
- Monitoring and debugging distributed workloads.

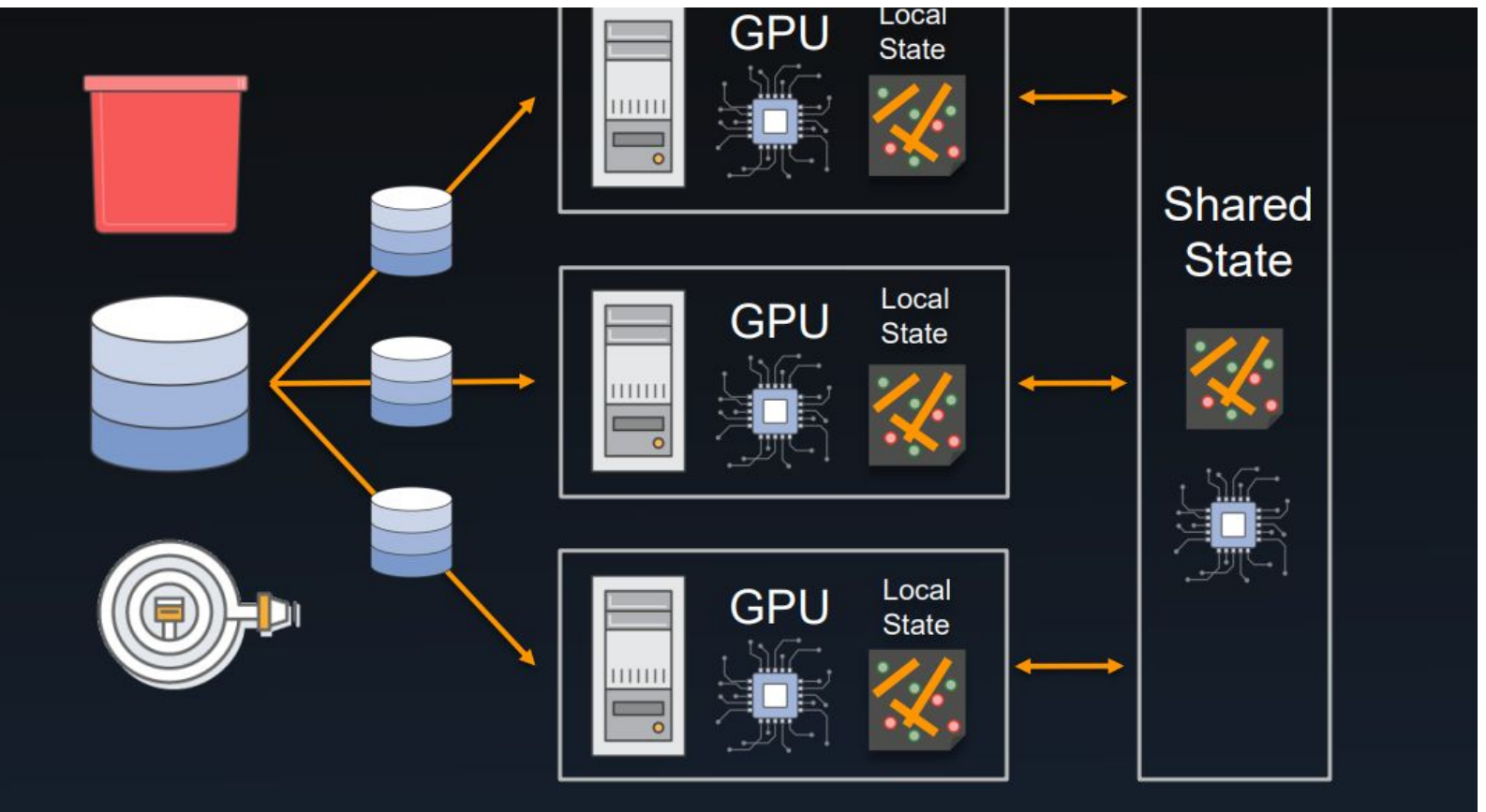
Risk of inefficiencies:

- Uneven resource utilization.
- Long runtimes due to poorly optimized parallelism.



Key Features of SageMaker Distributed Training

Scalability



Monitoring and Debugging

less than 20 seconds ago

SageMaker Debugger

Monitor and profile your training jobs in real time.

Monitoring Profiling

Configure profiling Stop training Download report

Overview Nodes

Training job summary

Time spent in phases of training

Total training time: 23m 47s

- Initialization: 3m 17s 94ms
- Training loop: 20m 30s 38ms
- Finalization: 130ms

Training job details

Start time	2020-10-19 13:43:12:579
End time	2020-10-19 14:06:59:580
Job duration	1427.000 seconds
Training loop start	2020-10-19 13:46:29:674
Training loop end	2020-10-19 14:06:59:711
Training loop duration	1230.036 seconds
Initialization	197.095 seconds
Finalization	0.131 seconds
Initialization (%)	13.81%
Training loop (%)	86.20%
Finalization (%)	0.01%

Training progress over time

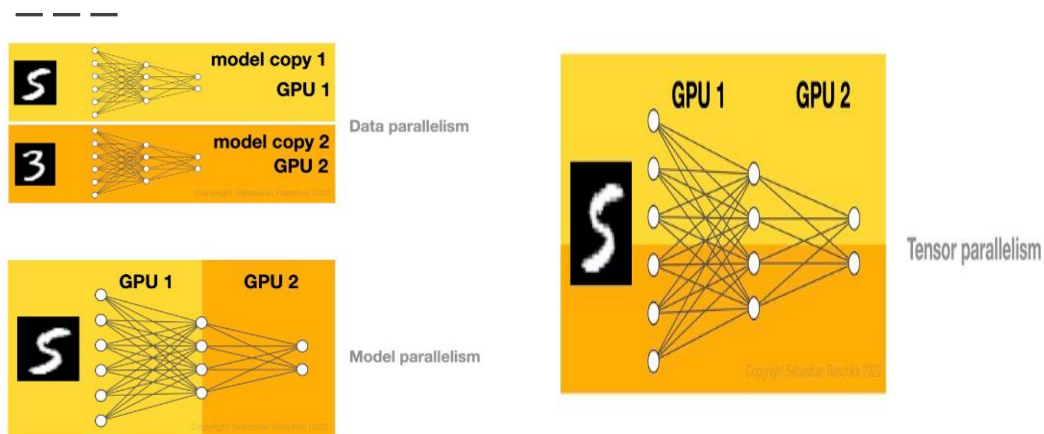
Insights

The following list shows a summary of Debugger rule analysis on your training job. Expand the following rule items to find suggestions and additional details, such as the number of times each rule triggered, the rule parameters, and the default threshold values to evaluate your training job performance.

Showing 8 suggestions

- GPUMemoryIncrease - Issue Found**
 Choose a larger instance type with more memory (if it is not a memory leak) or apply model parallelism
Number of times the rule triggered: 78
Number of violations: 78
Number of datapoints: 2855
Rule parameters:
 increase: 5%
 patience: 1000
 window: 10
 For more information, see the [GPUMemoryIncrease](#) rule description.
- BatchSize - Issue Found**
 Run on a smaller instance type or increase batch size
Number of times the rule triggered: 64
Number of violations: 64
Number of datapoints: 2854

Support for Different kinds of distributed training strategies



Access to AWS HPC Clusters enhancements

Real-World Use Cases

The researchers also conducted experiments in which they used SDP to train Mask-RCNN, a neural network with roughly 44 million parameters, on a computer vision task with about 118,000 training examples. The training time was six minutes and 45 seconds on PyTorch and six minutes 12 seconds on TensorFlow, [approximately 24% better than the previous record](#).

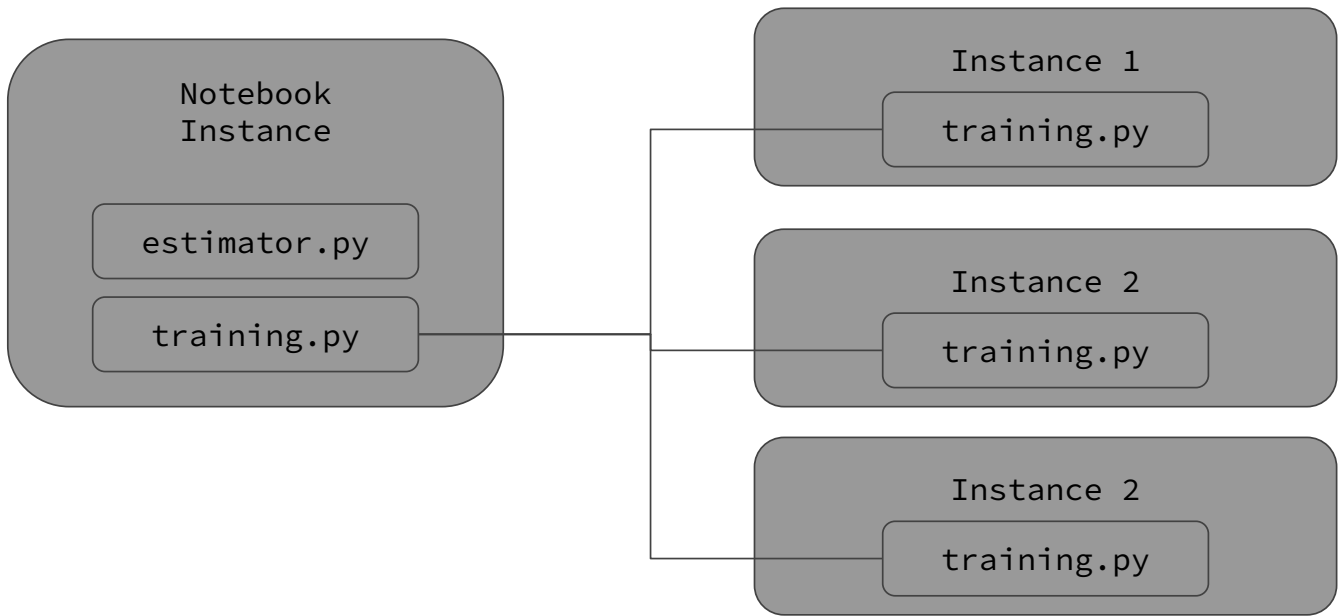
Example Use Case: Distributed Training with PyTorch

What is Distributed Training?

- Splits training tasks across multiple machines and/or GPUs.

SageMaker's Distributed Training Framework:

- Easily configure training clusters.
- Automatic management of resource scaling.



35

Steps Taken:

- Create a Training Script for the Distributed Job
- Create an Estimator to launch training script across instances
- Set training configurations (i.e. instance-type, preprocessor-worker count, etc.) in estimator.
- Launch from JupyterLab

36

Conclusion

37

Criticism of SageMaker

- Prototyping very difficult
 - Very expensive and time consuming to prototype
 - Rebuilds instances from scratch each launch
- Many libraries are proprietary to SageMaker
 - Vendor lock-in
 - Document is sparse for the libraries
 - Difficult dependency management
- High learning curve
 -

38

Comparison of Costs

39

Benefits of SageMaker

- Customizable
-

40

Questions?

41

Conclusion

Key Takeaways:

- Amazon SageMaker provides an end-to-end managed environment for building, training, and deploying machine learning models.
- SageMaker Studio is an interface for launching instances, viewing training jobs and accessing metrics.

42